Методичка 3.3 - Знакомство с отладчиком OllyDbg

Открытие программы

Отладчик OllyDbg позволяет подключиться к существующему процессу или запустить программу самостоятельно (Attach или Open).

Open - запуск программы под отладчиком
Attach - подключение к уже запущенной программе (существующему процессу)

Обзор интерфейса

Открываем под отладчиком программу prog-1.4.3.exe из прошлого урока. Рекомендую сразу подобрать оптимальный для себя шрифт во всех окнах (кликаем правой кнопкой мыши на любой области - Appearance - Font (all) - Font 6)

Отладчик OllyDbg имеет 4 основные области:

Окно дизассемблера

Окно дизассемблера расположено посередине и имеет самый большой размер. Это самое главное окно. Здесь находится дизассемблированный код программы, который мы сейчас отлаживаем. Слева расположены адреса, далее опкоды инструкций, сами инструкции и результат анализа инструкций, который выполняет OllyDbg. Анализ инструкций в отладчике включен по умолчанию, и при необходимости его можно отключить в настройках. Отключают в тех случаях, когда видно, что отладчик распознал код неправильно, например, представил код, как данные или наоборот.

Пример:

Правой кнопкой в окне дизассемблера - Analysis - Remove Analysis from module

Правой кнопкой в окне дизассемблера - Analysis - Analyse Code

Регистры процессора

Окно расположено в правой части окна. В этом окне можно видеть состояние регистров процессора в режиме реального времени. В самом начале идут базовые регистры, регистры, которые указывают на стек, индексные регистры, далее регистр EIP, регистры состояния и сегментные регистры.

Пример:

Можно выполнить несколько команд (F7) и будет видно, как меняется значение в регистре EIP.

В названии окна отладчика можно видеть на какой модуль сейчас указывает регистр EIP.

Сейчас регистр EIP указывает на модуль prog_1_4, т.е. это секция кода программы, но стоит перейти в функцию MessageBox, модуль поменяется на USER32. Это значит, что регистр EIP сейчас указывает на код в библиотеке user32.dll.

Стек

Окно расположено внизу справа. В этом окне мы видим состояние стека для нашего процесса. Со стеком связаны регистры ESP и EBP. Первый указывает на вершину стека, а второй на начало стекового фрейма. Если нажать правой кнопкой на стек, то можно выбрать Go to EBP или Go to ESP.

Пример:

Перезапускаем программу - Ctrl + F2.

Выполняем несколько инструкций PUSH перед вызовом функции MessageBox и видим, что количество значений в стеке увеличивается и регистр ESP всегда указывает на вершину стека.

Адресное пространство процесса

Здесь можно наблюдать состояние адресного пространства процесса, т.е. памяти процесса. Можно посмотреть содержимое памяти по любому адресу. Для этого достаточно нажать правой кнопкой и выбрать Go to … (Ctrl + G) и ввести нужный адрес. Если по указанному адресу памяти не будет, то отладчик нам сообщит об этом, написав, No memory. В этом окне удобно анализировать данные, которые расположены в куче или в секции данных.

Обзор функциональности

Было рассмотрено 4 основных окна, но есть еще и другие окна. Они могут быть вызваны по кнопкам, которые доступны на верхней панели.

Кнопка - E (executables).

В этом окне можно увидеть перечень динамических библиотек, которые используются программой и узнать по какому адресу они были загружены в памяти. Мы видим, что программа prog-1.4.3.exe была загружена по базовому адресу 0x00400000.

Библиотеки ntdll.dll, kernel32.dll и kernelbase.dll всегда подключаются к любой программе в ОС Windows.

Библиотека user32.dll была подключена, так как мы используем функцию MessageBox.

Кнопка - M (memory).

В этом окне можно удобно анализировать адресное пространство процесса.

Здесь можно видеть стек программы, PE-заголовок, секции программы, а также все библиотеки, которые были подключены к программе.

Также в этом окне можно выполнять поиск строк в памяти. Правой кнопкой - Search (Ctrl + B).

Кнопка - C (cpu).

Возвращает к главному окну, которое называется CPU. Оно же окно дизассемблера.

Кнопка - / (patches)

В этом окне можно найти инструкции, которые были изменены в процессе отладки. Сейчас это окно пустое. Давайте попробуем изменить код в окне CPU.

Кликаем правой кнопкой на инструкции PUSH 0 - Binary - Edit

Меняет байт 0x00 на 0x01 (6A 00 > 6A 01).

Снова нажимаем на кнопку - / и видим, что инструкция PUSH 0 была изменена на инструкцию PUSH 1.

Можно отменить патч.

Кликаем правой кнопкой на патче и выбираем - Restore Original Code (Space).

Кнопка - K (call stack)

В этом окне виден стек вызовов. Если открыть это окно, то оно будет пустым, потому что пока не вызывали функций внутри программы.

Далее можно выполнить несколько инструкций (F8) до вызова функции MessageBoxA и затем нажать - F7, чтобы перейти внутрь этой функции.

Видно, что адреса слева сильно изменились, так как теперь выполнение кода происходит в библиотеке user32.dll.

Можно снова открыть окно, где отображается стек вызовов и увидеть, что был вызов функции MessageBoxA и параметры.

Возвращаемся к окну CPU и выполняем еще несколько инструкций (F8) до вызова следующей функции - MessageBoxTimeoutA и снова нажмём F7, чтобы перейти внутрь функции.

Откроем снова окно, где отображается стек вызовов.

Добавился еще один вызов функции MessageBoxTimeoutA и параметры.

С помощью стека вызовов можно отслеживать полную цепочку вызова функций до любого места в программе. Это может быть удобно, чтобы понять, как программа дошла до определенного места в коде.

В правом нижнем углу мы можем видеть статус программы: Running, Paused или Terminated.